Digital System Design

Professor Neeraj Goel

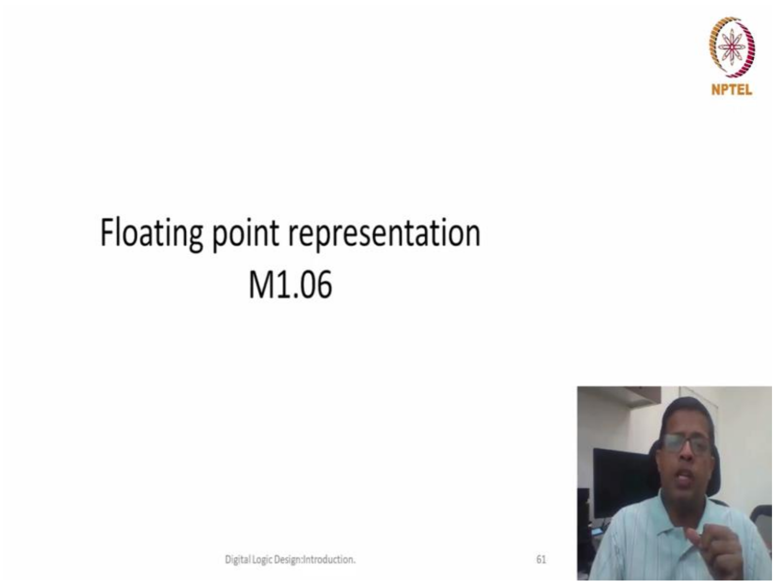
Department of Computer Science Engineering

Indian Institute of Technology, Ropar

Lecture 8

Floating Point Number - 1

(Refer Slide Time: 00:17)



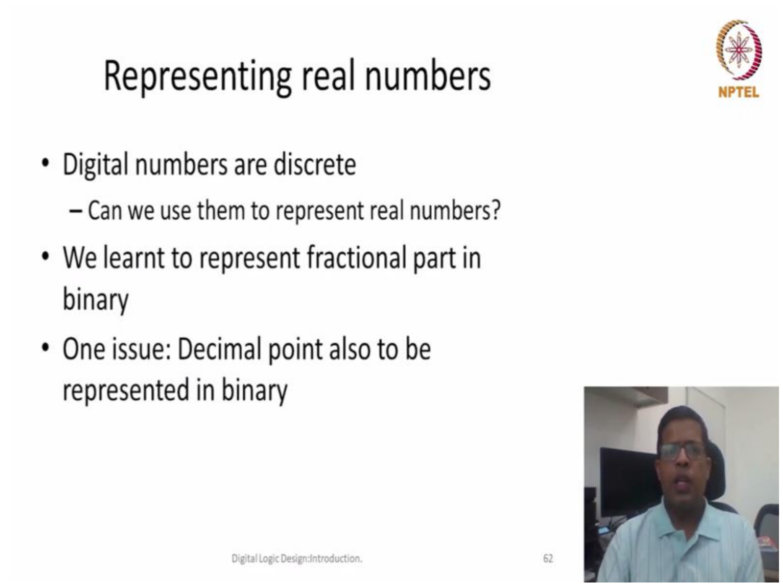
Floating point representation

M1.06

Digital Logic Design:Introduction. 61

Hello everybody in our previous class we discussed various different type of notations BCD number representation in binary or text representation or any other symbol representation. So, today what we are going to discuss is can we represent a real numbers in binary or not?

(Refer Slide Time: 00:45)



Representing real numbers

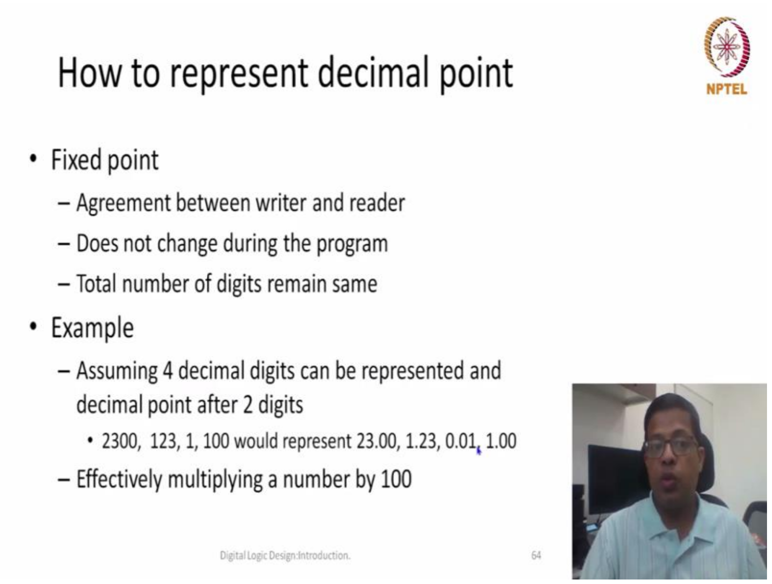
- Digital numbers are discrete
  - Can we use them to represent real numbers?
- We learnt to represent fractional part in binary
- One issue: Decimal point also to be represented in binary

Digital Logic Design:Introduction. 62

In our first lecture, we understood that digital numbers are discrete. So discrete means they are there they are not continuous. Now, can we use these discrete numbers to represent continuous numbers like a real numbers? In word around us actual numbers are both integers as well as real numbers. Integers means counting 12345 but real numbers were fractions are also there.

Now, can we represent numbers that contained both fractional part and integer part or essentially, we call them real numbers. So, can we refer to it using binary and we represent real numbers? Using of course yes, because in even in our one of the previous class, we have seen that how a fractional part can be represented we took various examples and you also gave in class quiz around that. So, that means that can be done. Now, but the major point again the major point here is or the most difficult point which you have to understand and see is that how can we put in this decimal point?

(Refer Slide Time: 02:17)



How to represent decimal point

- Fixed point
  - Agreement between writer and reader
  - Does not change during the program
  - Total number of digits remain same
- Example
  - Assuming 4 decimal digits can be represented and decimal point after 2 digits
    - 2300, 123, 1, 100 would represent 23.00, 1.23, 0.01, 1.00
  - Effectively multiplying a number by 100

Digital Logic Design:Introduction. 64

You see in, in binary as we have seen earlier that this decimal point we can only have 0s and 1s. Now, this decimal point is also has to be represented either using 0 or 1 or we have to fix a point we have to fix the precision to say that, only 10 digits after only attend this decimal point will appear only after 10 digits. So, this decimal point has to be represented. Now, this fixed point or one way of representing this decimal point is that we do not write decimal point per se, but we made an agreement between a reader and writer.

Person who is always giving the binary numbers and who is reading this binary number. Now, when this agreement is there, that means we are we are saying that there is a point which exist. So, let us take this with an example. So, let us say we say we have a 4 decimal digit number. And we say that the point will always exist only after two points. So that means let us say we have a number 2300.

And because in a fixed point notation, we said that the decimal point will exist after two digits. It is not written over there. But it is implicitly assumed because there is an agreement, this agreement between a reader and writer whoever is writing this number, for example, you are writing 2300, because there is an agreement that decimal point has to be considered after two digits.

So that means this number, whoever is reading he will understand the number is 23.00. Similarly, if 123 is given, then because decimal point would exist, after two points after two digits, then it will be 1.23. That would be it would be read as 1.23 although it is written as 123 but because of that agreement that there is a point which exists but which is invisible you do not see that.

So, this could be one way of representing a decimal point without writing a decimal point. I would again like to rephrase this question also what question we are solving? Because decimal point also need to be represented as 0 or 1 we have only two symbols in decimal system. In decimal system, we invented one more symbol. So, initially there were 10 symbols 0 1, 2, 3, 4, 5, 6, 7, 8, 9 then, because we wanted to represent we wanted to write a real number so, we invented one more.

One more symbol called dot. So, because of this dot, now, there are a total of 11 symbols because of this dot we are able to represent real numbers also, but in case of binary, we do not have that choice. So, this decimal point either we assume that it exists or we have to use 0 or 1 for representing decimal. So that that, that thing is a challenging thing that how we can represent this decimal point.

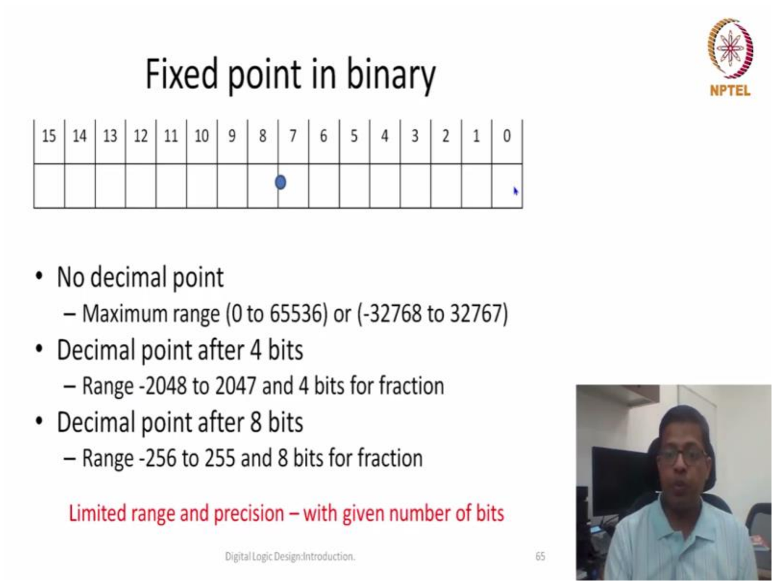
So, the representation where we are assuming this assumption, let us say we are taking an assumption that this point decimal point occurred at this particular digit place, then you cannot violate that agreement, all the numbers who are following that agreement will always put the decimal point at that place that cannot change with the program. So, let us say your program is executing for one hour or maybe for multiple days, then this representation of fixed point will always remain for all the numbers.

And yeah, so total number of digits will also remain same. So, these are the these are the some constraint, which are imposed on fixed point. So, if you look at the other way, so let us say you are looking at this number 23.00. So, this 23.00 is effectively multiplied by 100. And then return as 2300. So, whenever I am writing, I am multiplying because of decimal is exists, exists after two digits, so I am multiplying it by 100.

If decimal point exists after three points, three digits, then I would pay it with 1000. So effectively, whenever I am writing, I am multiplying it with some particular sector. And whenever I am reading, I am reading 2300. So, I would divide it by 100. So, I will get that 23.00 is the actual notation or if number 1 is given, then I had to divide by 100. So, it will become 0.01.

So, in this way, we can represent binary numbers also, these are the example of decimal numbers in the similar way, in binary numbers also, we can represent this point. Now, let us take an example of binary numbers.

(Refer Slide Time: 08:10)



Fixed point in binary

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								●							

- No decimal point
  - Maximum range (0 to 65536) or (-32768 to 32767)
- Decimal point after 4 bits
  - Range -2048 to 2047 and 4 bits for fraction
- Decimal point after 8 bits
  - Range -256 to 255 and 8 bits for fraction

Limited range and precision – with given number of bits

Digital Logic Design:Introduction. 65

So, let us consider that we have a 16-bit number, this number could be actually 32 bit number or any number, but let us say we are fixing the size or number as 16 bit number. For sake of completion, we will see that the maximum range of the 16-bit number would be 0 to 65,536 is the number is considered as completely positive, it is not the it does not have signed representation, only unsigned representation.

And if we are representing sign numbers also then in two's complement the range is minus 20 32 to plus 32,000. So, this is the range, if there is no decimal point, if the number is either completely positive or it has some negative part also. Now, let us say we have a decimal point. So, where we will put decimal point that is also a question. So, let us assume that we are putting a decimal point after 4 binary digits or essentially after 4 bits.

So, after 4 bits if we are assuming there is a decimal point again try to understand it is only an assumption, assumption means there is no explicit this this, this binary point is implicit and it would be understanding of whoever is reading and whoever is writing. So, let us say my binary point is assumed at this location where 4 bits that means 0123 will form the fractional part and bits number 4 to 15 will form the will the magnitude part.

Now, this means, means that I have reduced the number of bits representing magnitude from 16 to 12. So, that also means that the range will be reduced, I can only represent if it is a, this is a sign number that means, I can only representing instead of 32,000 I can represent only 2000 minus 2000 to plus 2000 numbers and only 4 bits of fractions. 4 bits a fraction means that 1 of the 16 levels so, whatever is the fractional part we will divide them into 16 parts and we will see that which 16 part we would like to take.

So, that means this number is very precision is very less, so, after decimal only if a convert into, into a decimal number, then effectively only 1 digit after the decimal point, 1 decimal digit after 1or 1decimal point too less. So, if I want to increase the precision, I have to move this decimal point to let us say some other place, let us say after 8 bits. If these 8 bits after inputting this decimal point after these 8 bits, the, the range of number is further squeezed, I can only represent from minus 256 through 255 any less.

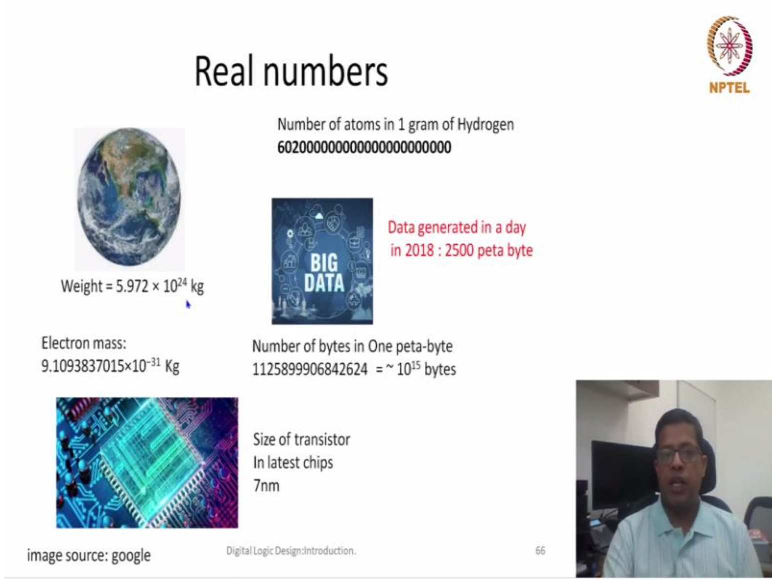
And the fraction is also 8 bit which is also less. So, that means a fixed-point representation could be a good possibility. But there are two or there are multiple issues. One issue is that it will reduce the range of numbers which I can represent. And also, precision is not that great, the amount of like a number of bits I can use for fractional part would be severely limited by the size of my number.

And because I would always if I want to fulfill here, the smallest number I can represent is 1, which means that it would be 0.00000001. So, the size of number is also quite less, it depends on the number of bits I am using. If I increase the number of bits, that that also does not solve the problem much. So that means there has to be some, some better solution. Still, these fixed-point numbers are our preferred in at least some part of computation, they are very important, we cannot simply ignore them.

So, we will see later, if we use some other method, then the computation addition, subtraction and multiplication is, is a very complex operation. But here, I can assume all of these numbers fixed point numbers as integers and do all the computation assuming they are integers, and then some correction factor can be applied over it after it. So, that that is the biggest advantage of this fixed point numbers that you, you might have heard that, like in a lot of computational programs, like in machine learning and all.

So initially, all the floating-point numbers or that complex format, which we are going to discuss in our next slide, we will use what because it was computationally so heavy that people said we will lose the accuracy, but because computation time would be less so, we can move towards fixed point format. So, that means a fixed fight format in a sense can give us a faster computation, but with very less accuracy, because precision is very less number of bits that can be represented after decimal is on fasting it is fixed the other thing they are also limited and small in number. So, that means we need to find a some other method to represent these real numbers.

(Refer Slide Time: 14:24)



Real numbers

Number of atoms in 1 gram of Hydrogen
602000000000000000000000

Weight = 5.972×10^{24} kg

Data generated in a day in 2018 : 2500 peta byte

Electron mass: $9.1093837015 \times 10^{-31}$ Kg

Number of bytes in One peta-byte
1125899906842624 = ~ 10^{15} bytes

Size of transistor
In latest chips
7nm

image source: google

Digital Logic Design:Introduction. 66

And also, there is one more reason to represent these real numbers. So, these real numbers are not only represented just to have this fractional part and, and magnitude part, but also they help us in representing very, very large numbers. So, for example, if we see what is the mass of Earth? Weight of Earth? It is approximately 5.972 into 10 is to power 24 kg, quite large. Now, if I want to use I want to consider this as a integer number, if I want to consider this as an integer number a number of bits, I would require just to represent this number would be more than 80.

So, that means, the format which will have 32-bit integers or 64 bit long, they will not be able to contain this large number I have to have something which would be for example 128 bits to represent this number. So, to represent large numbers and this is one of the large number there could be so, many things around us which, which represent quite a large number. So, for example, if you see these another example from your chemistry if you see number of atoms in 1 gram of hydrogen, you call it 1 mole Avogadro’s numbers 6.02 into 10 is to power 23.

So, many 0s are there, so, that if I want to represent them, even if I want to represent them in, in integers that means number of bit required would be large. So, there are other examples of even bigger numbers which are like in the world of computation, previously, we were only talking about megabyte then gigabyte then terabyte, then now, we are talking about petabytes. So, in petabytes, petabytes 10 is 415 bytes are there and to, to see this magnitude that it is not a number which is unrealizable which is not being used.

So, if we see look around us, then the amount of data which is generated in one day is 2.5 quintillion or 22500 petabytes that was in 2018. So, that was two years back now, we are doing much more Facebook we are doing much more YouTube. So, the amount of data which is generated in big data schemes, because we are using clouds because we are using the social media and because there are IoT nodes which are very small and so many of them are there each of them are generating a lot of data. So, the amount of data generated is huge.

Now, we have to represent we have to sometime count that or sometime we have to write that data somewhere. So, that is not only huge storage is required, but also we need to represent those large numbers. And it is not the question of only large numbers, but there are small numbers also. So, for example, the size of transistors in the latest VLSI chips is around 7 nanometer seven nanometer means 7 into 10 is to power minus 9 quite small.

And similarly, we see the electron mass or anything which is related to quantum the size is quite less, number is quite small. So, that means the, the summary of this slide is I would like to represent very, very large numbers like 10 is to power 50 10 is to power 30 also, I would like to represent very, very small numbers like massive electron. So, I have two problems in hand. Now, first, I would like to have a scheme of representation, which can represent a very large number as well as very small number.

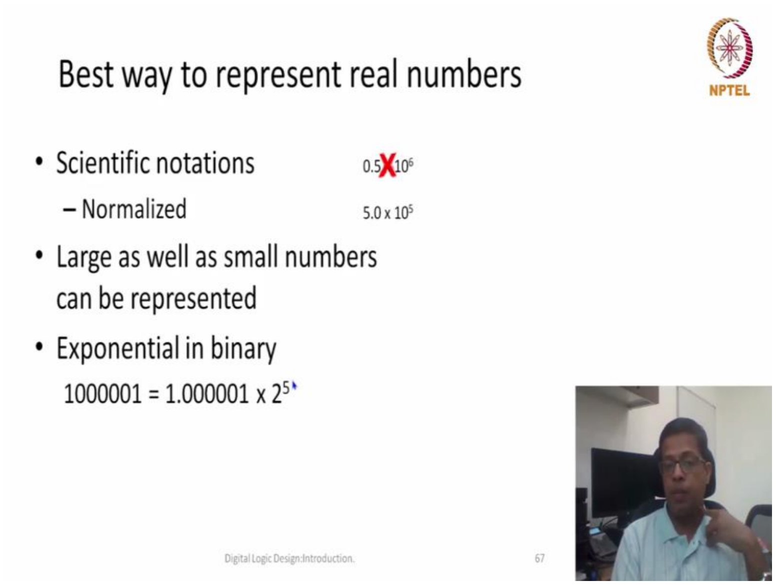
The second thing is the decimal point should be represented in such a way that it is dynamic in nature. It cannot be fixed point like the previous representation of fixed point, where I need to have lots of memory, lots of bits to represent one small number. Because I can always put spare more number of bits, but later I have to do computation also. And also, you see let us say this

this large weight, we have not written the rest of the 21 digits here because we see they are insignificant.

And we are worried only about the significant digits and we have taken only 4 significant digits here. So, although this number of atom here return is 0, so I am not sure whether it is actually 0 or it is approximated. Yes, so what I mean to say that we would like to represent these numbers, but not in absolute sense, but in a relative sense also it will work. So, how to represent such kind of a large numbers with the point which is not fixed, we call that Floating Point, means the decimal point is not at a fixed location, but it would move along with the computation.

So, how do we do that? How suppose that something is coming in your mind. Yes, we use scientific notation as it is written over here. This is a scientific notation that you have, you are writing some number then into multiplication, and then you are, you are writing an exponential symbol. So, the way we are doing it in decimal system, in the similar way, we can do the same thing in a binary system also.

(Refer Slide Time: 20:49)



Best way to represent real numbers

- Scientific notations $0.5 \cancel{x} 10^6$
  - Normalized 5.0×10^5
- Large as well as small numbers can be represented
- Exponential in binary

  $1000001 = 1.000001 \times 2^5$

Digital Logic Design:Introduction. 67

So, in a binary system, if I would like to do then I have to use the scientific notations which are normalized. So, let us spend some time on understanding what is normalized. Although all of you understand that but let us this is for the revision sake, that let us say this is a number in a decimal system 0.5 into 10 is to power 6. Is it normalized? No, a normalized number will have a non-zero significant only 1.

1 digit before decimal point, before decimal point there has to be one non-zero digit, and after decimal point, there could be anything any number of digits could be there, but before decimal there has to be one non-zero decimal digit. So that means this number has to be written as 5.0 into 10 is to power 5. Why normalize forms? Because otherwise there would be so many of these notations and all of them would be equivalent.

So that means this the same number can also be written as 15 to 10 is to power 4 or 500 into 10 is to power 3. So, to avoid all of those different forms, which are actually infinite in nature, rather than that, we say only one standard form in a standard or a normalized form, there would be only 1 non 0 digit before decimal and after decimal, in the right of decimal, there could be any number of digits, which could be 0 or 1.

If we are writing 0, that means they are showing precision, if there if so, that means if I were writing 0, there are also significant in nature, because they are showing the precision. Now, in case of binary, how it would happen in binary also, there would be like for example, this is the number 1.001000001. So, in this number I want to represent as a exponential form or a scientific notation, then it has to be 1.000001 into 2 is to power 5.

Because 1 this has to be 6, 2 is to the power 6. So, we can correct it. So, this is actually 2 is to the power 6. Now, this particular way of representing a real number, satisfy most of our requirement with a small exponent is not a very large exponent. So, the examples which we have seen in our previous slide, they were talking about 10 is to power 20 10 is to power 30 as the largest numbers and there that is actually the largest number.

Now, those because it is an exponential form, I would require very less number of digits a very small number here to represent that. The other thing it can represent a very small number as well as very large number large number means the exponential power would be in positive in a very small number that means exponential power would be negative.

So, I can represent very large number a very small number as well as I can control the precision, precision could be could be fixed I know that how many precisions which would are there. In fixed point number that was not the freedom I have. I only had like how many digits I can have here, I can precisely choose or I can choose at how many precision bits I can keep in this normal, normal, normal notations or scientific notations.

So, this is a this looks like a good way to move forward and represent a Floating-Point Number but again, we are asking this question multiple times, but this question would be asked again here that now how we will represent it in binary, we do not have a notion, we do not have a symbol to represent dot. And here we are introducing two more symbols in our decimal system one is the Multiplication another is the Exponent.

So, this luxury was there in decimal system that we can keep on introducing new symbols. So, here we have in addition of decimal, we have these two new symbols basically, I would require not only in 9 10 symbols have 0123456789, but also require these three additional symbols of Point, Multiply and Power. So, what is the alternative? So, this, this alternative is the question which we have to answer.

And if we are able to answer that question, that means we can represent our, our numbers. Yeah, so additionally one more thing, it is not only these three, but there is also two more which means whether this, this, this particular number could be positive or negative. And similarly, this exponent could also be positive and negative. So that means I need to introduce plus as well as minus so five symbols are there. So, I need to represent these five symbols in binary, then I would be able to do it represent these exponential notations in, in binary.